

---

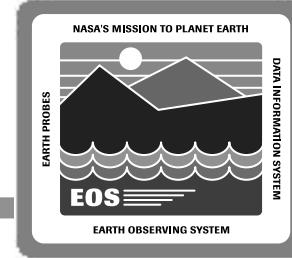
# PAS Flight Operations Tools

## Tony Cetuk

---

17 October 1995

# PAS Flight Operations Tools Overview

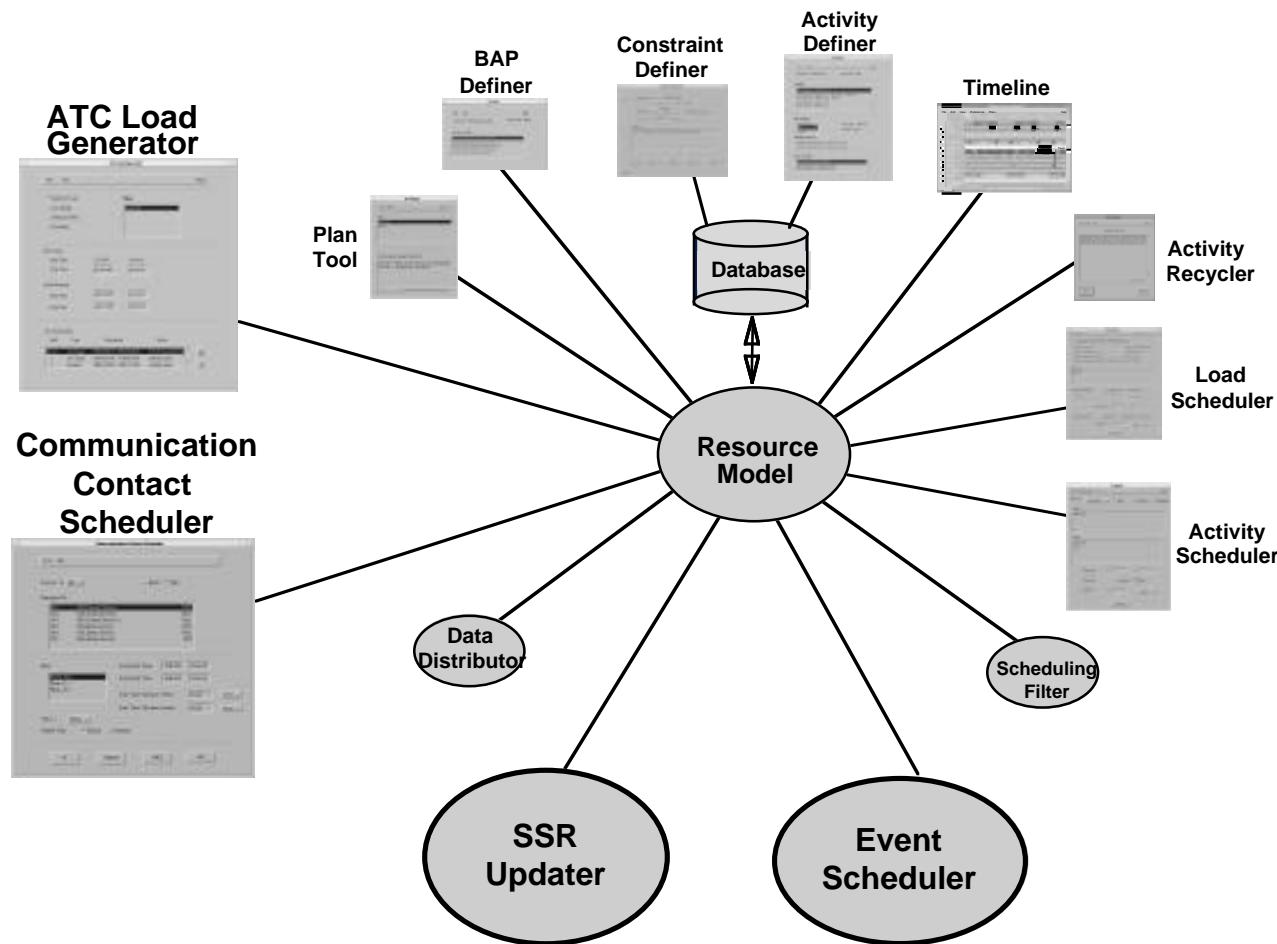
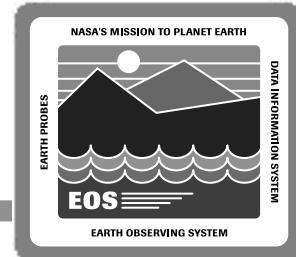


**PAS Flight Operations Tools** are generally used at the EOC to assist the FOT/IOTs in schedule development.

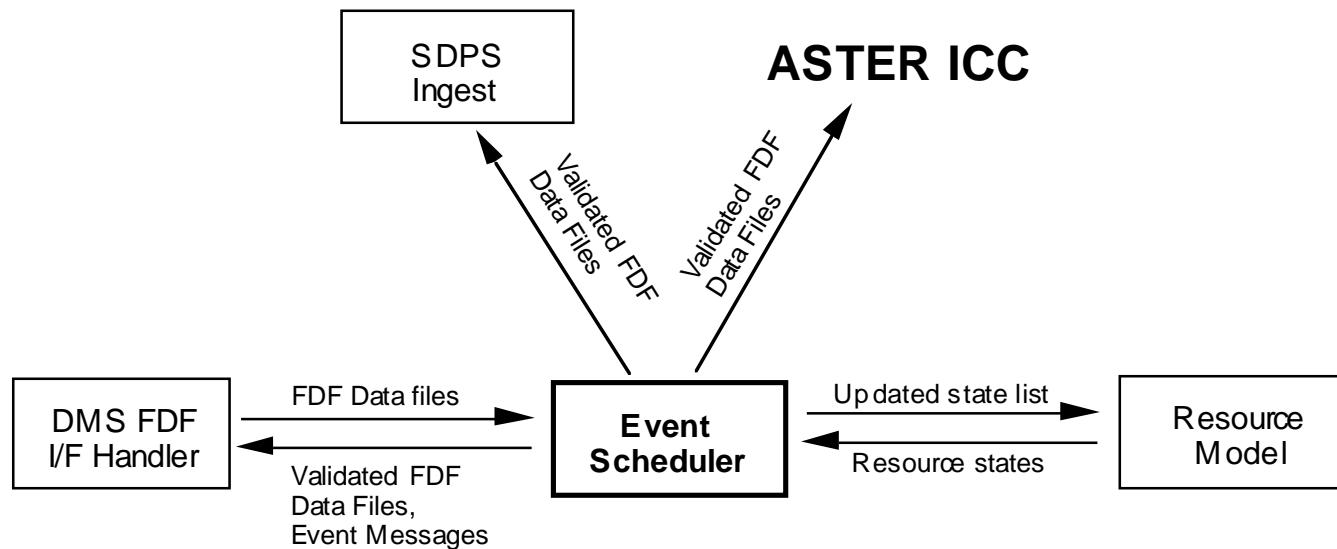
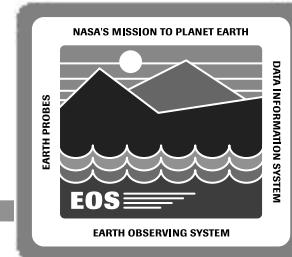
**PAS Flight Operations Tools include:**

- Event Scheduler
  - Updates the mission plan based on new FDF orbit data
- Communication Contact Scheduler
  - Establishes NCC communication contacts (TDRSS or ground) to support recorder playbacks and uplink activities
- ATC Load Generator
  - Creates an ATC load based on the integrated, conflict-free schedule
- SSR Updater
  - Updates the SSR data volume model based on “as-flown” Analysis information

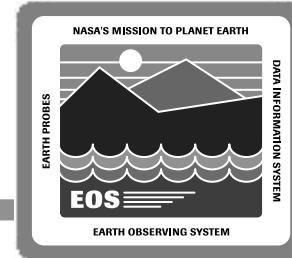
# PAS Flight Operations Tools



# Event Scheduler Context



# Event Scheduler Design



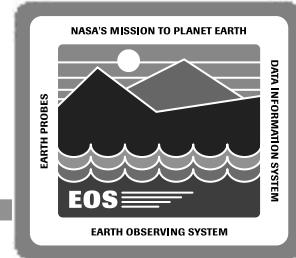
**Performs quality assurance of FDF orbital data**

- Validates data content based on product header information
- Performs continuity checks between products to ensure data integrity

**Builds EOC-specific planning aids to support mission needs**

- Derives EOC-specific planning aids from FDF orbital data
  - Computes CERES sun-avoidance times from FDF Sun azimuth, elevation and beta angles
  - Computes MISR Local Mode times from FDF ground track data

# Event Scheduler Design (cont.)



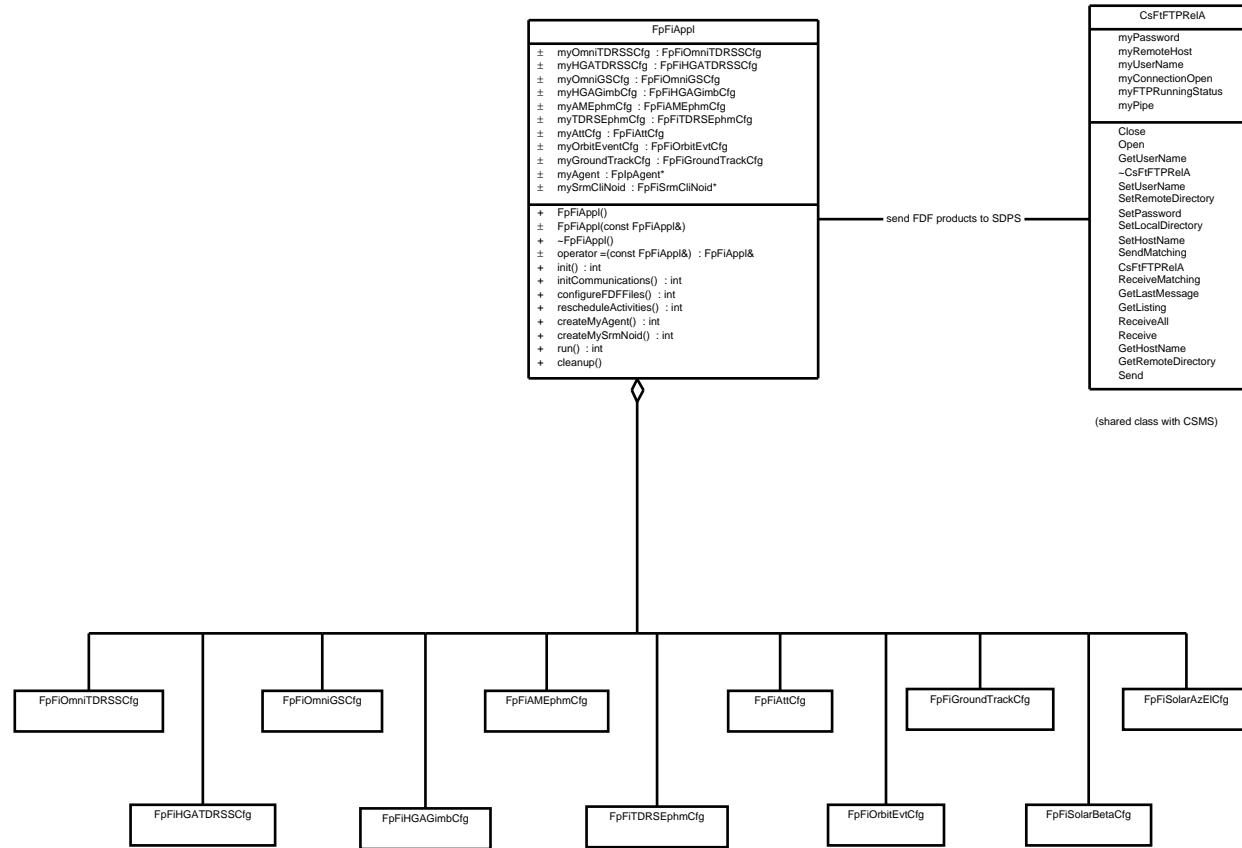
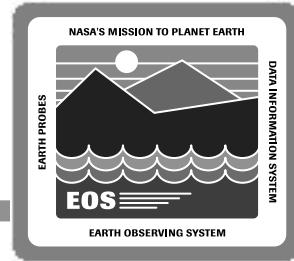
## Evolvable design for adding new planning aids

- Future FDF event types can be added as a software component
- System provides algorithmic flexibility for building EOC-specific products

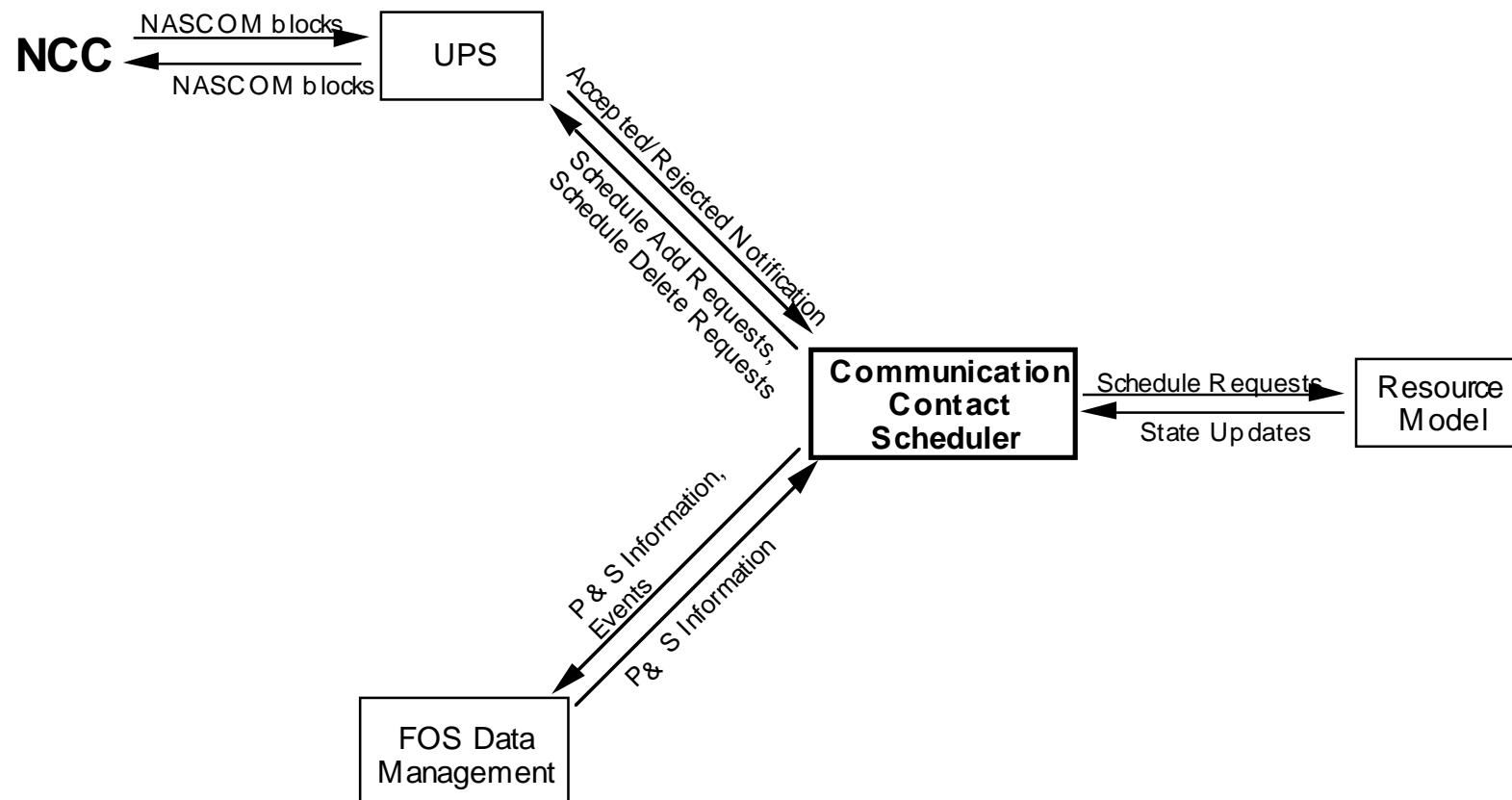
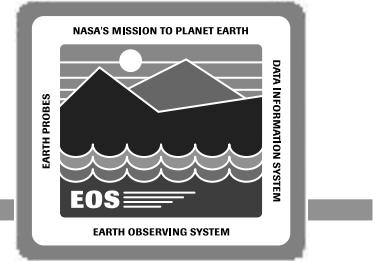
## Automatically updates mission schedule based on refined orbital data

- Updates the “event pool” to reflect the latest event times
- Re-schedules all activities impacted by modified event times
- Incorporates additional orbit data for extending the scheduling period
- Distributes planning aids to SDPS and ASTER ICC

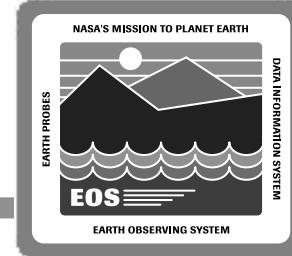
# Event Scheduler Object Model



# Communication Contact Scheduler Context



# Communication Contact Scheduler Design



**Uses electronic interface of UPS for NCC communication**

- UPS handles the transmission and reception of NASCOM 4800 bit blocks

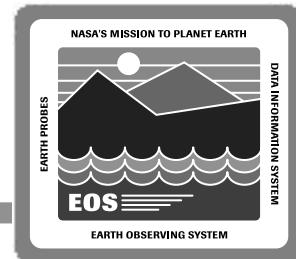
**Supports communication contacts on NCC space and ground resources**

- Nominal contacts scheduled on TDRSS
- Contingency communications scheduled on ground stations (DSN, WOTS, GN)

**Allows FOT to establish communication contact requests with NCC**

- Utilizes pre-defined prototype ids and configuration codes
- Allows user to define their own prototype ids for unique communication contact needs

# Communication Contact Scheduler Display



Communications Contact Scheduler

File SAR

Resource ID HGA

Batch  SAR

Prototype IDs

A01	MA Forward Service	NCC
A02	SSA Forward Service	NCC
A03	KSA Forwrd Service 2	User
B01	MA Retrun Service	NCC
B02	SSA Return Service	NCC
B03	KSA Return Service	NCC

Plans

Master Plan  
What-if\_1  
What-if\_2

Event Start Time 1999/12/31 12:00:00

Event Stop Time 1999/12/31 12:11:00

Start Time Tolerance (Plus) 00:03

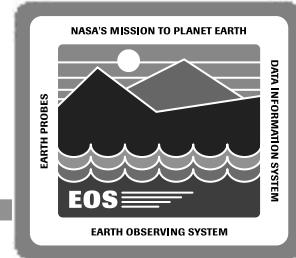
Start Time Tolerance (minus) 00:01

TDRS Id TDRS-E

Support Flag  Normal  Premium

OK SCHEDULE CANCEL HELP

# Communication Contact Scheduler Design (cont.)



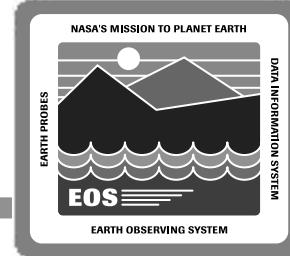
**Provides two modes of scheduling**

- **Batch scheduling**
  - Automatically computes communication contacts based on optimization strategy
- **Manual scheduling to incorporate individual activities**

**Automatically updates the mission schedule based on information received from UPS**

- **State of communication contacts changed from:**
  - “Scheduled” to “Confirmed”
  - or “Scheduled” to “Rejected”
- **Creates summary report of confirmed or rejected contacts**

# Communication Contact Scheduler Optimization Strategy



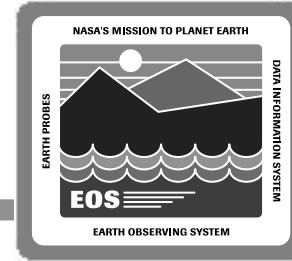
**Every potential contact yields a score based on a depth-first search that assigns values to communication contact criteria**

- **Timing criteria include:**
  - Time duration of contact
  - Time interval between two contacts
- **Scheduling criteria include:**
  - Data volume in each SSR buffer
  - Primary TDRSS/Ground Station availability

**Algorithm controlled by database defined parameters to alter selection criteria**

- **Parameters include look-ahead depth, granularity and threshold**

# Optimization Strategy Object Model



```

FpCsScore
myRanges
+ FpCsScore()
+ FpScore(const FpCsScore &)
+ ~ FpCsScore()
+ operator =(const FpCsScore &)=FpCsScore &
+ calculateScore(): int
+ doLookup(float): int
+ doLookup(const char *) int
+ addRange(HObject *)
+ rangeIter()
+ calculateScore(FpCsInView*, FpCsInView*)int

FpCsCommLinkScore
+ FpCsCommLinkScore()
+ FpCsCommLinkScore(const FpCsCommLinkScore &)
+ ~ FpCsCommLinkScore()
+ operator =(const FpCsCommLinkScore &)=FpCsCommLinkScore &
+ calculateScore(FpCsInView*, FpCsInView*)int
+ get(istream &): int
+ put(ostream &): int
+ calculateScore(FpCsInView*, FpCsInView*)int

FpCsConstScore
? myMinVal: int
? myMaxVal: int
+ FpCsConstScore()
+ FpCsConstScore(const FpCsConstScore &)
+ FpCsConstScore(int,int)
+ ~ FpCsConstScore()
+ operator =(const FpCsConstScore &)=FpCsConstScore &
+ calculateScore(FpCsInView*, FpCsInView*)int
+ get(istream &): int
+ put(ostream &): int
+ minVal(int): void
+ minVal(): int
+ maxVal(int): void
+ maxVal(): int

FpCsDataScore
+ FpCsDataScore()
+ FpCsDataScore(const FpCsDataScore &)
+ ~ FpCsDataScore()
+ operator =(const FpCsDataScore &)=FpCsDataScore &
+ calculateScore(FpCsInView*, FpCsInView*)int

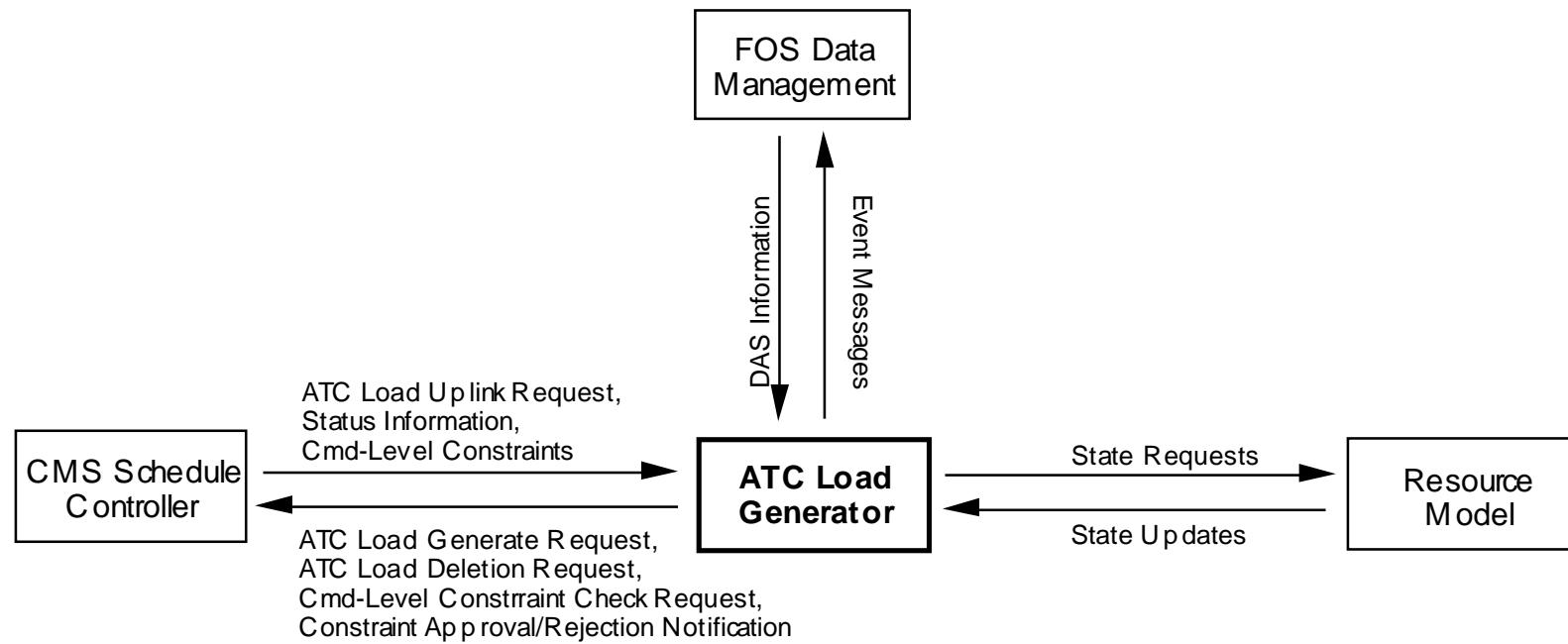
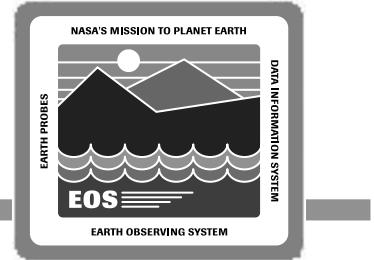
FpCsHGASlewScore
+ FpCsHGASlewScore()
+ FpCsHGASlewScore(const FpCsHGASlewScore &)
+ ~ FpCsHGASlewScore()
+ operator =(const FpCsHGASlewScore &)=FpCsHGASlewScore &
+ calculateScore(FpCsInView*, FpCsInView*)int
+ get(istream &): int
+ put(ostream &): int
+ tdrsAOI(const HString &)HEpochTime
+ tdrsLOS(const HString &)HEpochTime
+ computeSlewTime(HEpochTime, HEpochTime)
+ calculateScore(FpCsInView*, FpCsInView*)int

FpCsDurationScore
+ FpCsDurationScore()
+ FpCsDurationScore(const FpCsDurationScore &)
+ ~ FpCsDurationScore()
+ operator =(const FpCsDurationScore &)=FpCsDurationScore &
+ calculateScore(FpCsInView*, FpCsInView*)int

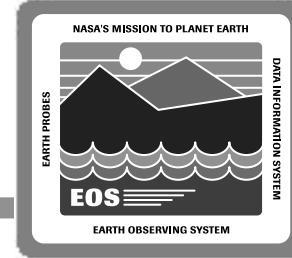
FpCsSeparationScore
+ FpCsSeparationScore()
+ FpCsSeparationScore(const FpCsSeparationScore &)
+ ~ FpCsSeparationScore()
+ operator =(const FpCsSeparationScore &)=FpCsSeparationScore &
+ calculateScore(FpCsInView*, FpCsInView*)int

```

# ATC Load Generator Context



# ATC Load Generator Design



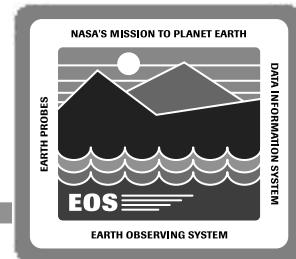
**Provides FOT with capability to generate an ATC load**

- **Freezes mission plan to allow no further modifications over ATC load time period**
- **Checks activity-level and command-level constraints**
  - **Verifies no hard constraints exist**
  - **Allows FOT to approve or disapprove soft-constraints**
- **Re-generation of ATC loads to support late changes**
- **Simulation loads can be generated to support mission tests that involve the spacecraft simulator**

**Provide FOT and IOTs with analysis-mode**

- **Activity and command-level constraints can be checked without generation of ATC load**

# ATC Load Generator Display



**ATC Load Generator**

File    Jobs    Help

◆ DAS/ATC Load  
◆ Late Change  
◆ Constraint Check  
◆ Simulation

**Plans**

Master Plan

**DAS Times :**

Start Time	3/7/1999	00:00:00
Stop Time	3/8/1999	00:00:00

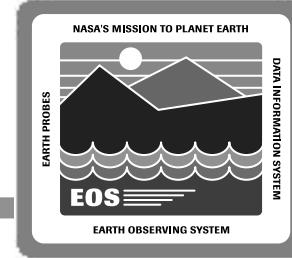
**Uplink Request :**

Start Time	3/7/1999	10:00:00
Stop Time	3/7/1999	16:00:00

**Jobs Information :**

Job #	Type	Time Interval	Status
Current	Late Change	3/5/99 00:00:00 - 3/6/99 00:00:00	15:32:00 Generating DAS
1	Late Change	3/6/99 00:00:00 - 3/7/99 00:00:00	Waiting in queue
2	Simulation	3/9/99 15:30:00 - 3/9/99 17:30:00	Waiting in queue

# ATC Load Generator Design (cont.)



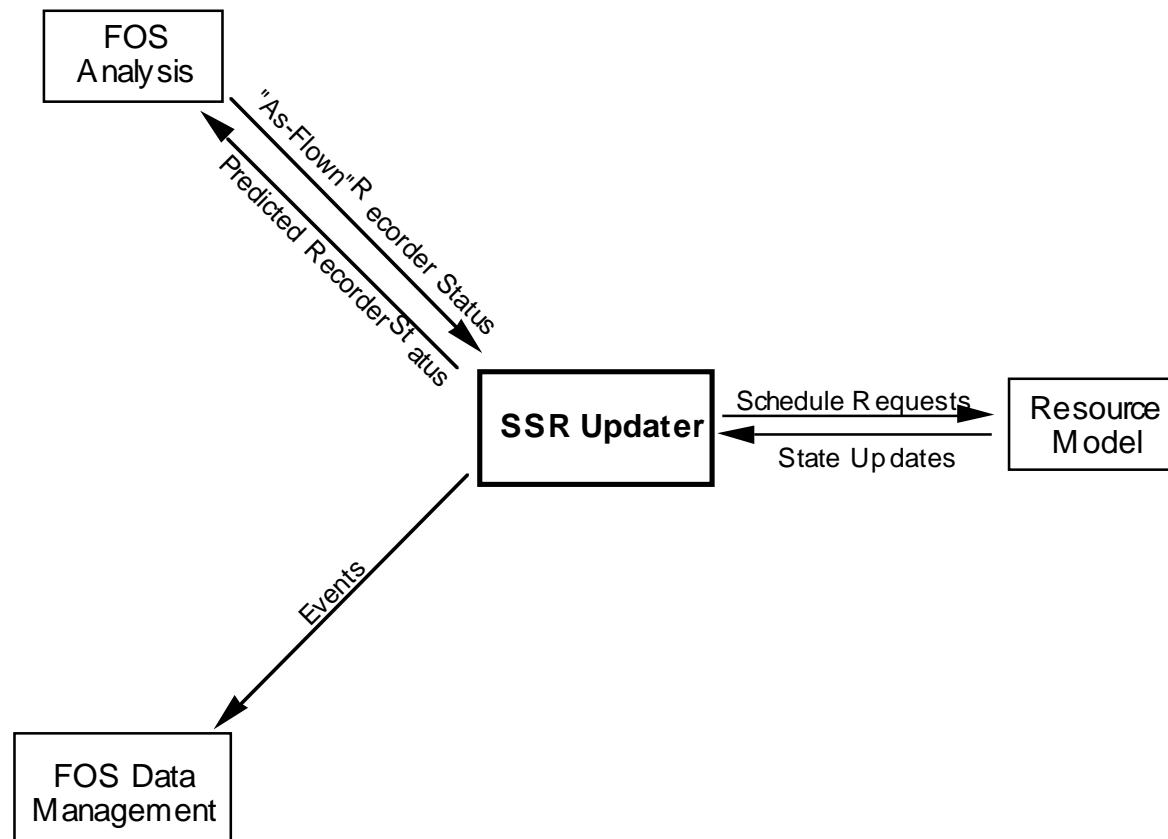
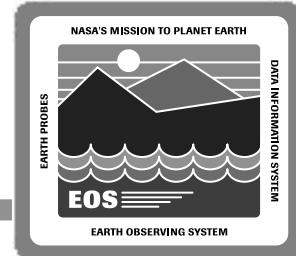
## Job Status Queue

- Provides status messages throughout load generation
- Supports multiple jobs in processing queue

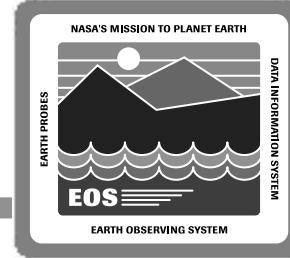
## Automatically determines ATC load uplink times

- Based on user requested time window
- Selected uplink times can be modified by FOT

# SSR Updater Context



# SSR Updater Design



**Automatically updates the SSR data volume model based on “as-flown” Analysis information**

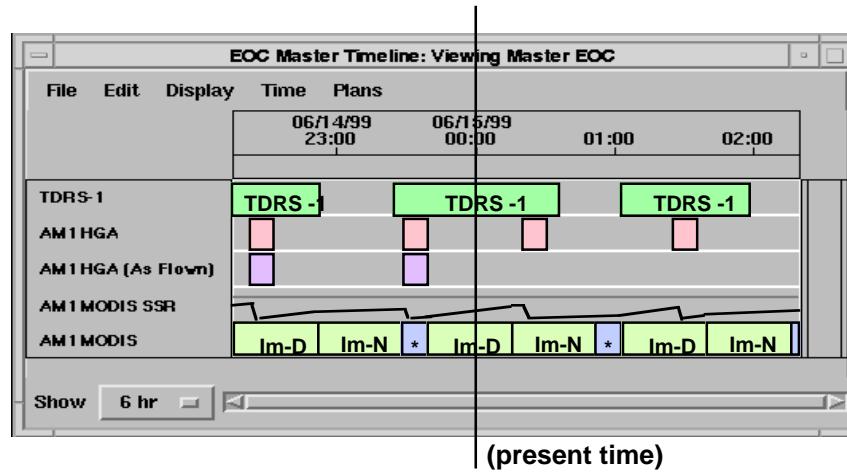
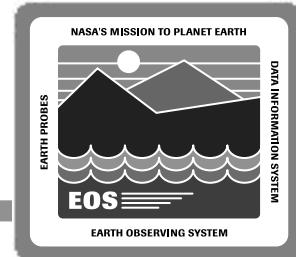
- Data volume updated after each communication contact (TDRSS or ground station)
- Notifies the user of data overflow constraints
- Updates mission plan to reflect accurate SSR data storage
  - Timeline reflects modified storage estimates

**Interfaces with SSR model for providing FOS Analysis with:**

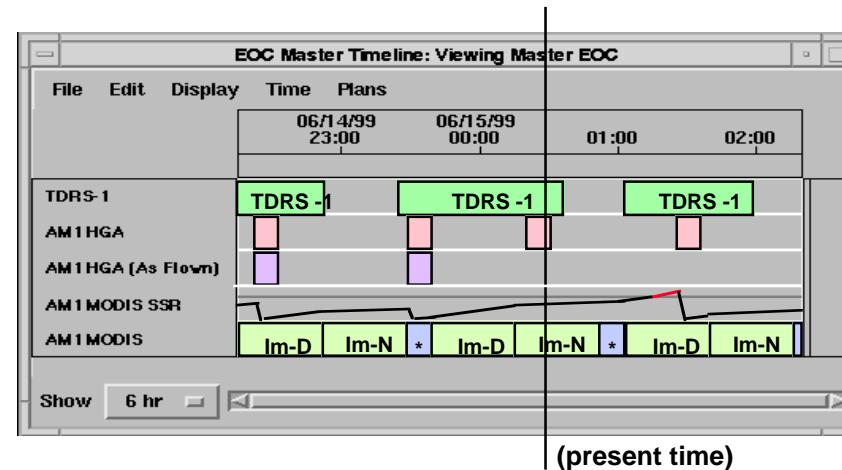
- Predicted buffer counter and data volume
- Predicted contact times and playback duration

# SSR Updater

## “As-Flown” Timeline Updates



**MODIS SSR Buffer at 00:00  
(No deviations)**



**MODIS SSR Buffer at 00:30  
(Missed TDRSS contact)**